Bachelor's Thesis

Degree Programme in Information Technology

Bachelor of Engineering

2010

Jere Keltamäki

BITTORRENT PROTOCOL

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT

UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Information Technology

04.03.2010 | 38

Instructor Patrick Granholm, Kalliopi Skarli

Author Jere Keltamäki

Bittorrent protocol

The Bittorrent protocol has become the dominant peer-to-peer file distribution protocol in the recent years, with over 160 million client applications installed worldwide. The main objective of this thesis was to find out what makes Bittorrent such an effective file distribution method and to give detailed descriptions how the protocol works in practice, while also looking at some disadvantages that the protocol has.

The information presented in the thesis was gathered by studying in depth different pieces of research conducted about the different aspects of the Bittorrent protocol and through the author's own knowledge acquired from years of using the Bittorrent as a file distribution instrument.

Through the in-depth explanations of the Bittorrent operations, the thesis forms a solid view of the protocol and shows that the protocols success comes from the Tit-For-Tat (TFT) strategy implemented in it and from the ability to distribute the file transfer responsibility from one single host to multiple hosts. By sharing the transfer responsibility, the Bittorrent protocol eliminates the problem of single point of failure, which is of great importance in the file distribution production networks.

KEYWORDS:

Bittorrent, peer-to-peer, P2P, file distribution, protocol

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka

04.03.2010 | 38

Ohjaaja Patric Granholm, Kalliopi Skarli

Tekijä Jere Keltamäki

BITTORRENT PROTOCOL

Bittorrent on vertaisverkoissa käytettävä tiedonsiirtoprotokolla, joka on viime vuosien aikana saavuttanut huiman suosion. Tämän opinnäytetyön tarkoitus on tutkia Bittorrent protokollaa ja selvittää mikä aiheuttaa sen suuren suosion tiedonsiirtovälineenä, ja antaa yksityiskohtainen kuvaus siitä kuinka protokolla toimii käytännössä ja minkälaisista ongelmista se mahdollisesti kärsii. Opinnäytetyön materiaali koostuu Bittorrent -protokollasta tehtyjen monien eri tutkimuksien ja tekstien tulkinnasta yhdistettynä usean vuoden omakohtaiseen kokemukseen ja tietämykseen protokollan käytöstä.

Opinnäytetyön yksityiskohtainen kuvaus Bittorrent -protokollasta osoittaa, että protokollan menestys tehokkaana tiedonsiirtovälineenä johtuu pääasiassa siinä käytettävästä Tit-For-Tat (TFT) strategiasta, joka takaa tiedonsiirron reiluuden. TFT-strategia yhdistettynä Bittorrent -protokollan kykyyn jakaa tiedonsiirtovastuu yhdeltä käyttäjältä usealla käyttäjälle luo pohjan vakaalle verkolle, joka pystyy tarjoamaan nopeita tiedonsiirtonopeuksia.

ASIASANAT:

Bittorrent, vertaisverkko, P2P, tiedonsiirto, protokolla

CONTENTS

1 INTRODUCTION

Bittorrent is a peer-to-peer (P2P) file distribution protocol that differs from the traditional client–server distributions in the Internet, in such way that the active participating users help each other in the file distribution, thus eliminating the single point of failure that exists in the traditional client–server schemes. In this way the capacity of the protocols throughput increases as the number of users participating increases, allowing Bittorrent to perform highly scalable file distribution. Since the release of Bittorrent there has been many pieces of research published regarding the different functions of the protocol, but none of them really give an in-depth overview of all the different functions and how they work as a whole. The aim of this thesis is to study all the mechanisms that make the Bittorrent protocol so effective and popular file distribution method, while also giving some examples of disadvantages and abuses that the protocol might suffer from. The goal is to create the thesis so that it is understandable to people who are not familiar with Bittorrent protocol, while also giving an in-depth explanation of all the functions to those who want to get a deeper understanding of the protocol. The thesis explains the current technologies used in Bittorrent and gives a glimpse of where the development of the protocol is heading.

This thesis is organized as follows: Chapter two gives a general overview of the Bittorrent protocol, Chapter three explains the functions of the Bittorrent protocol in detail, Chapter four explains the differences between Bittorrent file distribution and traditional client–server file distribution, Chapter five explains the different attacks and abuses that the Bittorrent protocol suffers from, and what kind of countermeasures can be taken against them, and Chapter six is the summary.

2 BACKGROUND

Bittorrent is a peer to peer (P2P) file distribution protocol that works over TCP and is often used for transferring large amounts of data over network. The protocol was designed by a programmer called Bram Cohen, who released the first version of the implementation on July 2001 [1]. Since the release, Bittorrent has become an extremely popular way of sharing data over the Internet, with over 160 million client applications installed worldwide [2]. The number of the installed client applications include all the different versions and variations available on the Internet; the most popular being Bittorrent Mainline, Azureus and µTorrent. Figure 2.1 shows the main window of the Bittorrent Mainline client application version 6.3. It has been estimated that as of February 2009 the Bittorrent protocols portion of the whole Internet traffic was approximately 27 – 55 %, depending on the geographical location from where the readings were taken. [1]

Although Bittorrent is widely known as a method for illegal distribution of copyrighted material between end-users, it has started to gain popularity also among different companies as a low-cost and effective way to distribute files. The creator of Bittorrent, Bram Cohen, is trying to lead the way and show that Bittorrent is more than just a way of distributing illegal material. Cohen formed his own company Bittorrent Inc. and made a deal with the Motion Picture Association of America (MPAA) not to share links to illegal material in the website bittorrent.com and turned it into a store that sells online video content [3]. In addition to the Bittorrent Inc., there are also a few other well known companies that use Bittorrent for the distribution of their files, such companies include Blizzard Entertainment, which uses Bittorrent to distribute their files and updates for the massive online multiplayer game World of Warcraft and Valve Software, which has gone as far as hiring Bram Cohen himself to help them with their Steam software distribution system. [4]

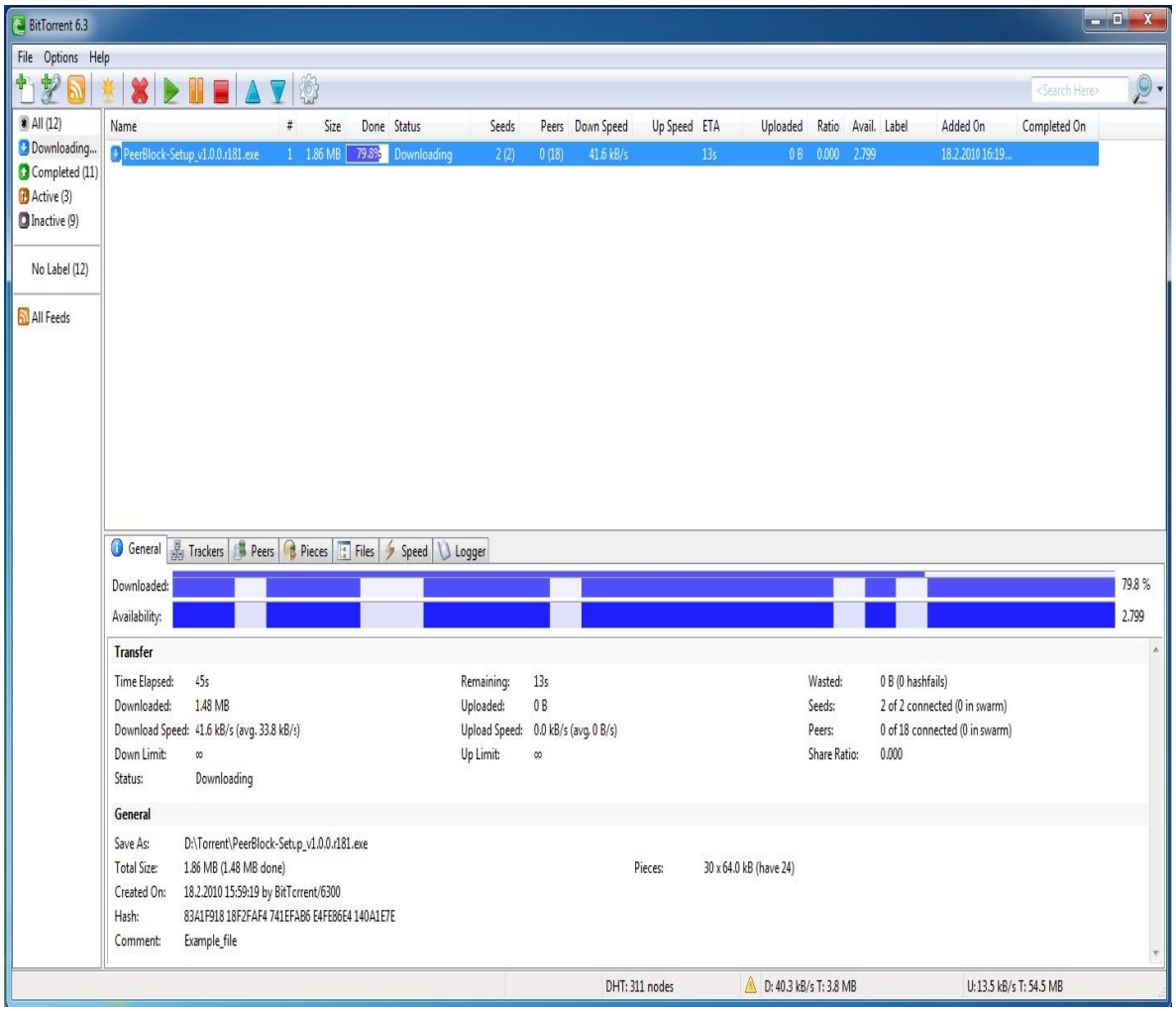

Figure 2.1 Illustration of the main window in Bittorrent Mainline client application (version 6.3). The window holds general information about the torrents being downloaded.

The way that Bittorrent protocol differs from typical file distribution methods such as FTP and HTTP, is by reducing the strain that is put on a single host server when large amounts of data are being distributed simultaneously. In a standard client–server scheme, a centralized server hosts a file and clients connect to the server to download the whole file. As the number of simultaneous downloaders increases, so does the strain put on the servers

upload capacity, and if the number of downloaders becomes too great the server is simply unable to distribute the file, or the transfer times become too long [1]. In Bittorrent protocol the centralized server only coordinates the connections between the peers; it does not have any knowledge of the actual contents of the files that are being distributed. In this way the server is able to handle a large number of connections with relatively small bandwidth utilization. The actual file is being transferred between the Seeders (uploading peers) and the Leechers (downloading peers). To make this possible, the file that is being transferred is broken in to small pieces, so that when a leecher acquires a piece of the file it can start uploading the piece to other leechers immediately, thus acting as a server and client at the same time, which is the main idea behind the Bittorrent protocol. In this manner the load of sharing the file to everybody who is interested in it is reduced from the initial Seeder and offloaded amongst all the Leechers. [2]

3 Operations of Bittorrent protocol

To start the file distribution in Bittorrent protocol, a static file with the extension .torrent needs to be created and uploaded to an ordinary web server. After that, the person who has the complete file, usually called initial seed, needs to make the file available by opening the .torrent -file with a Bittorrent client application and leave the application running. The client application should be left running at least as long as all the pieces of the file have been uploaded once. After the initial seeder has completed all the necessary preparations to distribute the file, the potential leechers can download the .torrent -file from the web server and open it with a Bittorrent client application. The HTTP links for the .torrent –files can usually be found from torrent indexing websites, such as isohunt.com. After the .torrent –file is opened with the client application; The client application will contact the tracker using the information in the static .torrent -file and use the peer information received as an response from the tracker to contact the initial seeder and start downloading pieces of the actual file. The following sections will discuss the methodology behind these operations in more detail. [5]

3.1 The .torrent file

The static torrent file is created from the actual file that will be distributed, and it can be created using a Bittorrent client application. The size of the torrent file varies from few kilobytes (kB) to few hundred kilobytes (kB), depending on the size of the actual file that will be distributed. The information that is included in the torrent file depends of the version of the Bittorrent protocol, but by convention the files have “announce”, “announce-list”, “nodes” and “info” sections [6]. The “announce” section specifies the URL of the tracker and the “info” section contains general information of the torrent file, such as name for the file, file length, piece size, piece count, private flag, file paths and directory name. The “announce-list” section adds support for the use of multiple trackers and the “nodes” section adds support for the use of DHT. [6]

The contents of the file are identified with an info-hash value, which is obtained by applying the SHA-1 hash algorithm to the “info” -section of the torrent file. By using this info-hash value the client application can verify that the downloaded file pieces are genuine and were not corrupted during the transfer, by comparing the hash values in the torrent file with its own hash values of the acquired file pieces. [6] Every received piece is first checked against the hash value of the corresponding file piece before proceeding with the download [5]. Certain information in the “info” –section can be changed without affecting the info-hash value, such as the trackers and comments. However, if any other information is modified the value of the info-hash changes and trackers see the torrent as a “new” torrent and they will not share the peer information of the unmodified torrent, even if the files are exactly the same. [7]

When the .torrent -file is opened with a Bittorrent Mainline client application, a window called “add new torrent” opens (see figure 3.1). In the window, a user can change the download directory and preview general information about the file, such as the name of the file, size of the file, date of the file, and possible comments left by the uploader. The window also contains an option to skip the hash check, but it is not encouraged to do, because then the data integrity of

the file is not verified after the download, and the retrieved file can be corrupted. [1] Through the “advanced…” -button the user can preview the “properties” –window. The “properties” window has two interleaves called ”General” and ”Advanced”. In the “General” -interleaf a user can preview and add tracker addresses, change bandwidth settings for the file, and choose other methods for peer discovery, such as DHT, Local peer discovery and Peer exchange (see Figure 3.2). In the “Advanced” –interleaf the user can preview and add web seeds, and define programs to run after the download has finished (see Figure 3.3). A user can change these settings or choose to accept the defaults provided by the .torrent -file, depending on the users own preferences.

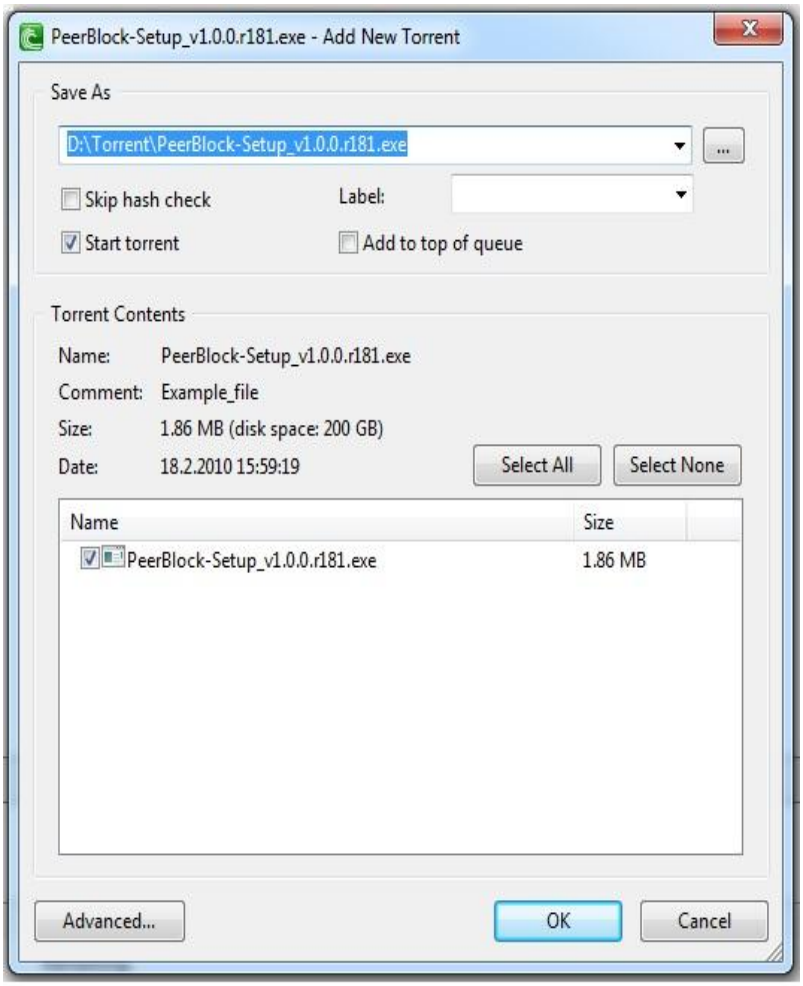

Figure 3.1 Illustration of the "Add New Torrent" –window presenting general information of the torrent to be downloaded in the Bittorrent Mainline client application.

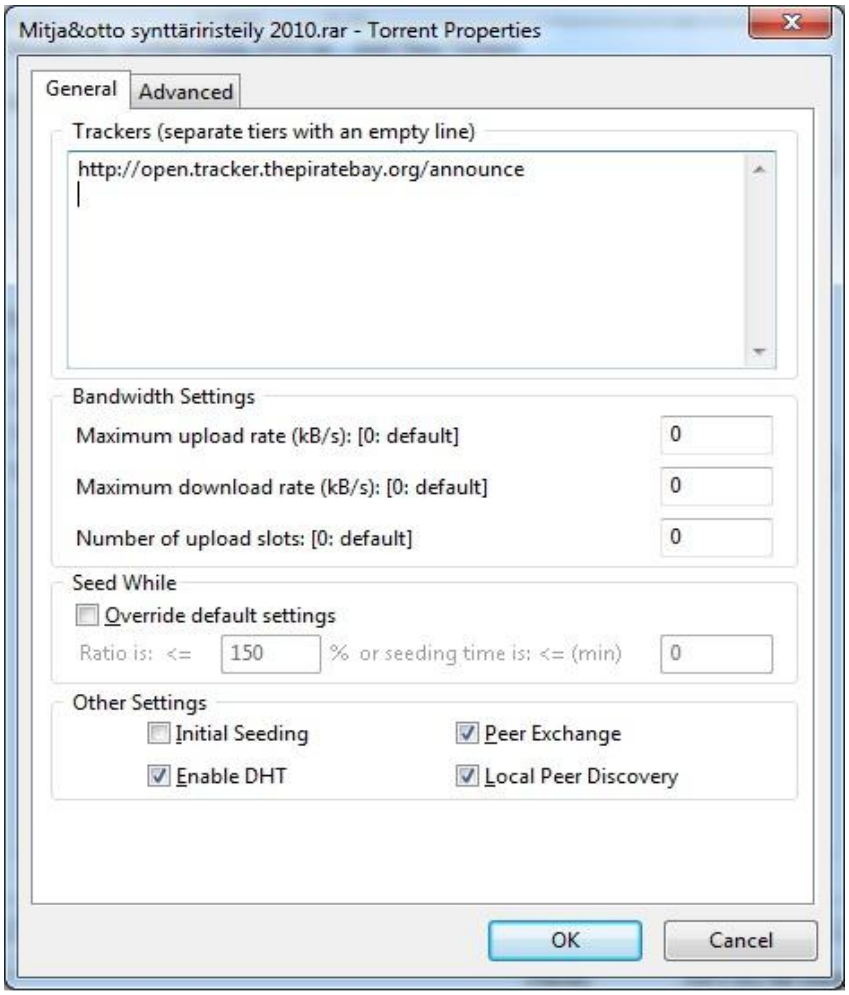

Figure 3.2 Illustration of the “General” –interleaf in the “Torrent Properties” –window, where the peer discovery and bandwidth settings for the torrent can be changed in the Bittorrent Mainline client application.

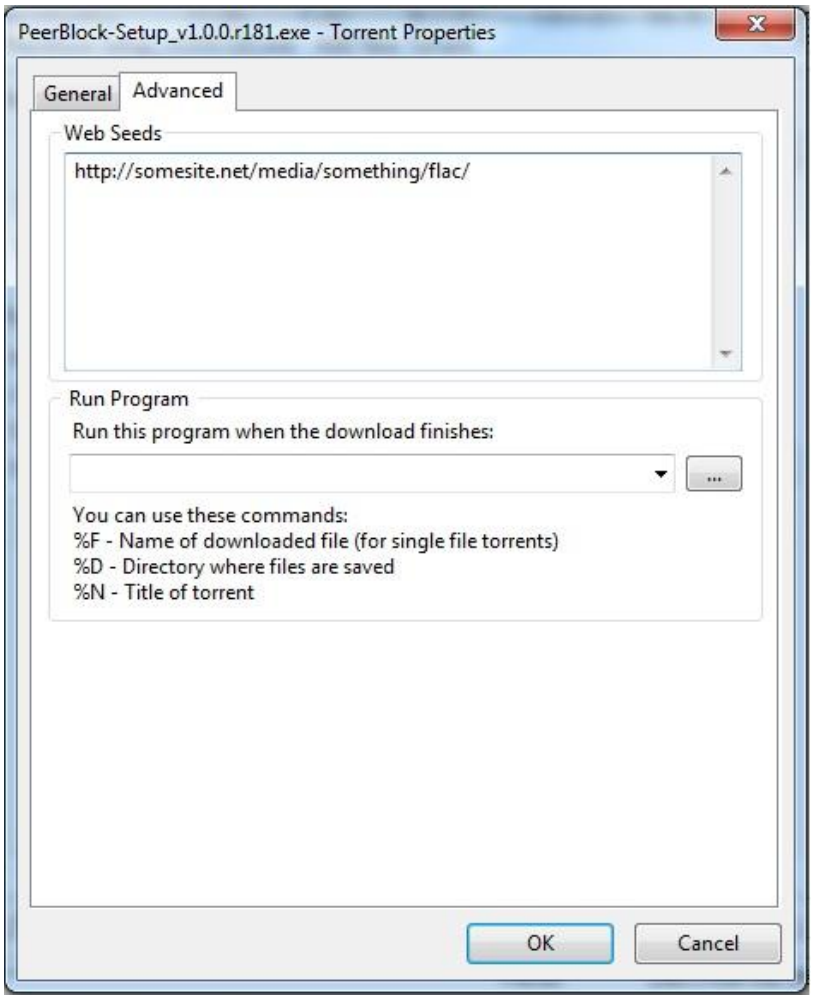

Figure 3.3 Illustration of the “Advanced” –interleaf in the “Torrent Properties” -window where web seeds can be added in Bittorrent Mainline client application.

3.2 Magnet links

Magnet links offer the torrent indexing websites an alternative way to advertise files, without hosting the actual .torrent -files. The standard for magnet links was developed already in 2002, but it has not been properly adopted to the Bittorrent protocol until recently, when the torrent indexing website called The Pirate Bay

started using them, and soon followed the launch of the world's first magnet link-only Bittorrent indexer, TorrIndex. The magnet links are based on the evolving Uniform Resource Identifier (URI) standard that is mainly developed for P2P networks. Magnet links hold information only about the contents of the data that they link to, rather than holding information of the location where the file can be retrieved [8]. This makes it even more difficult for the authorities to accuse the website owners who provide the magnet links of illegal activity. In Bittorrent the magnet links hold the hash value of the .torrent -file, which the Bittorrent client applications can use to find peers that have the same .torrent –file and the actual contents of the file. This helps the torrent indexing websites to save bandwidth, when they can just calculate the hash values themselves and allow users to download the hash value, rather than the actual .torrent -file itself. The torrent hash value is passed as a parameter in the magnet link, and when a user opens the magnet link with a Bittorrent client application, the client application starts immediately seeking addresses of peers that have the same .torrent -file, without seeking the peer information from the tracker first. As the client application finds the peers, it connects to them and downloads first the .torrent -file, and then the desired content. [9] This eliminates the need to rely on a single server that stores and distributes the .torrent –files. In addition to the hash value, magnet links can also hold other information about the file, such as the name of the file and links to the trackers used by the torrent. Although magnet links provide the torrent indexing websites with the possibility not to host any .torrent -files in the website itself, the magnet links can not replace them completely. The .torrent -files hold crucial information to get the download started, and all that information needs to be available in the swarm. [9] Although magnet links can not fully replace the use of .torrent –files, the decentralization characteristic creates the possibility for the torrent indexing websites to stop using trackers as a way of peer distribution. However, without the use of trackers, the website operators do not have any means to track the download and upload ratios of the users, so it is very unlikely that the private torrent indexing websites, which utilize the use of sharing-ratio will abandon trackers altogether.

The disadvantage of magnet links, as opposed to the .torrent –files, is that they do not provide any additional information of the actual file. When the magnet link is opened with a Bittorrent client application, the program does not show the opening window where the user can preview information about the file and change some settings for the file, such as the directory where the file is downloaded. This also prevents the users from selecting just some specific files in a multi-file torrent. Since the technology for magnet links is still being actively developed, these disadvantages are considered to be temporary setbacks that will most likely be corrected in the future. [8]

3.3 Peer distribution

The peer distribution in Bittorrent can happen in multiple ways. When the protocol was first introduced to the public the only method of peer distribution was through a centralized server called a tracker. As the Bittorrent protocol became more popular, other ways of peer distribution were introduced, such technologies include the use of multiple trackers, Distributed Hash Table (DHT), Local Peer Discovery (LPD) and Peer Exchange (PEX). The use of all these technologies in conjunction with each other gives the best outcome when trying to find peers to connect with. [6] It is also possible to restrict the peer discovery to use just trackers, by implementing a rule to the “info” –section of the .torrent –file to inform the client applications to restrict the use of decentralized tracking, regardless of what the user wants. The rule is mainly utilized by private torrent indexing websites that operate a private tracker, so that they can limit the use of their tracker for the registered users only. [1]

3.3.1 Trackers

In Bittorrent protocol, trackers are responsible for helping peers to find each other. A Tracker is a centralized server that collects a peer’s IP address and port number to share with other peers connecting to the same swarm; it has no idea of the actual contents of the files which are being distributed [1, 5]. A swarm is a set of peers that are participating in a distribution of the same files.

When a user starts to download a file with a client application, the client application announces itself to the tracker by telling it what file it is downloading and what port it is listening. The Tracker processes the announcement and returns information about a small random subset of peers in the swarm. The subset usually consists of 50 peers, both seeders and leechers. The user then uses this information to connect to the peers in the swarm subset to obtain the pieces of the file it is downloading. [6]

When the Bittorrent protocol was first published it only supported the use of one tracker at a time. This is sufficient in terms of handling all the possible connections, but causes a problem in tracker availability [5]. If the tracker becomes unreachable for some reason, for example, if maintenance work is carried out on the server, a new peer is unable to obtain information about the swarm, resulting in not being able to connect to other peers and start the download process. The peers who are already participating in the swarm can still continue the file distribution even if the tracker becomes unavailable, because they already have active connections to other peers in the swarm. To address this problem, a support for multiple trackers was added to the protocol. The use of the multi-tracker feature allows multiple trackers to take care of the same content. The multi-tracker feature can be used for two purposes; load-balancing or backup. This new feature added a new section to the torrent file, called “announce-list”, which contains a list of lists of tracker URLs. The trackers in the same list exchange information about the peers they know, which creates load-balancing. In the load balancing scheme a peer randomly selects one of the trackers in the list and sends its announcement to it. The different lists of trackers are used for backup purposes; if a peer sends an announcement to the trackers in the first list, but does not receive a response, it moves to the second list and tries to contact the trackers in there, and then the trackers in the third list and so on. The peer repeats the same steps every time it sends an announcement. The trackers in different lists do not exchange peer information with each other. [6]

3.3.2 Trackerless system

As an extension for tracker systems where centralized server coordinates all the connections between peers, a trackerless system was introduced (also known as distributed tracker system and decentralized system). A trackerless system was developed as a solution for the tracker availability problem. In trackerless system all of the peers act as trackers. The trackerless protocol extensions consist of the use of Distributed Hash Tables (DHT), Peer Exchange (PEX) and Local Peer Discovery. [1]

The purpose of the Distributed Hash Tables (DHT) is to find peers that are downloading the same files, without ever communicating with the central tracker. When DHT is used, the client application listens to UDP ports for peer discovery, instead of the TCP ports that are used when peer discovery is performed with the help of trackers. The peer discovery in DHT happens through routing tables that the Bittorrent client application keeps for known good peers. A peer is considered to be good if it has replied to a query within the last 15 minutes, or a peer who has ever responded to a query and has sent a query back within the last 15 minutes. The client application builds the routing tables from the peers that are introduced by a tracker. This way the routing tables can be maintained through the download of regular torrents. However, in a situation where the Bittorrent client application is newly installed and has not ever been used, the client application does not have any peers in its routing table, and the contacts need to be included in the “nodes” -section of the downloaded .torrent –file. The contact can be the peer who created the torrent or the closest peers in his/her routing table. Every peer has a globally unique identifier known as the “node ID”, which is selected at random from the same 160-bit space, that the calculated info-hash values of the torrents belong to. [10] The info-hash value is obtained by applying the SHA1 hash function to the info section of the .torrent –file [6]. When a peer wants to find a torrent it contacts the peers in its routing table who have the IDs closest to the torrent info-hash value and requests them for the contact information of the peers who are currently downloading the same torrent. If the contacted peer knows about peers that are downloading the same

torrent, it returns the contact information in the response. Otherwise, the contacted peer returns the contact information of the peers in its own routing table that have IDs closest to the info-hash value of the torrent. The requesting peer will further use those to query about the possible peers downloading the same torrent. The same steps are repeated until the search is exhausted, or the requesting peer finds the correct peers downloading the same file. No tracker is required at any time, which provides a solution to the single-point-of-failure of a central tracker server. [10]

In Peer Exchange (PEX) extension a peer takes advantage of the other peers that it is connected to, by asking them a list of peers that they are connected to. Using the peer-lists it receives, the peer can find more peers that are downloading the same file and peers that might be able to offer it faster transfer rates for the file. Just like in DHT, the right files and peers are found by comparing the hash value provided in the .torrent –file with the hash values of other peers. The user has to have one active connection to a peer before PEX can be utilized, but when started, PEX is considered to find more genuine peers than DHT or trackers. [9]

In Local Peer Discovery (LPD) extension, the peers find each other by sending multicast messages to find Bittorrent clients on the local network. One of the goals of Local peer discovery was to minimize the traffic that goes through the Internet service providers (ISP) channels and utilize the high bandwidths that the local networks can offer. [9]

The introduction of the decentralization in Bittorrent has created the possibility for the Bittorrent client applications to function as a search engine for the torrent files, thus making the torrent indexing websites obsolete. The idea of not having any torrent indexing websites is still pretty distant, but the developers of Tribler Bittorrent client application have started to implement such characteristics to their program. The Tribler does not just share the torrent files amongst the users, but also has several built-in spam control and moderation options, that encourage the users to keep the network clean, which is absolutely essential for a fully decentralized system to work properly. [9]

3.4 File distribution

In Bittorrent protocol, the files that are being distributed are broken into small pieces. This is done so that it would be easier to keep track of what pieces each peer has. Each peer informs what pieces it has to all other peers in the swarm. The pieces are always cut into a fixed size, which is typically 256 kilobytes (KB). To keep the transfer rates always as high as possible, it is important to have a continuous data flow without interruptions. Bittorrent tries to ensure a continuous data flow by pipelining. In the pipelining process the 256 KB pieces are further cut into smaller blocks of 16 KB over the wire, and approximately 5 requests are always pipelined at once. Every time that a requested block arrives, a new request is sent. The blocks of a particular piece can be requested from different peers. The maximum number of data to be pipelined is selected as a value which can most reliably saturate most connections. [5]

Bittorrent uses a strict policy in piece selection to achieve good performance. The first rule is that once a block of a particular piece is requested, the rest of the blocks of the same piece need to be downloaded before a block of another piece can be requested. In this way, complete pieces are acquired quickly and efficiently. Another piece selection policy is the rarest first –policy. This means that when a peer starts the download, it will request the pieces that fewest of its own peers have. By asking the rarest pieces first the peer will make sure that it will have the pieces that most of its peers will need and it can start the upload process as soon as possible. The rarest first –policy also enforces the idea of peers acting as a seeder and leecher at the same time. In a situation where there is only one seeder, it is beneficial that different leechers request different pieces from the seeder. In this way the upload capacity of the seeder is not wasted on redundant downloads and the seeder can transfer more data out faster. Using the rarest first –policy, the leechers can request the pieces that have not been uploaded yet by the seeder, by checking what pieces its own peers have. [5]

An exception to the rarest first –policy is a random first piece –policy, which is applied when the download starts for the first time. When the download begins for the first time the user does not have any pieces to upload, and as it is crucial that every leecher contributes also to the upload process, the leecher needs to obtain a complete piece quickly. As the rarest pieces are usually present on a single peer, they would take much more time to download than pieces that are present on multiple peers. For this reason, the blocks of the first piece are downloaded from random peers, and as soon as the first complete piece is assembled the policy changes to rarest first –policy. [5]

Sometimes towards the end of the download, the transfer rates might slowdown and completion of the download is delayed. To prevent this from happening, Bittorrent goes to endgame mode. In endgame mode, the Bittorrent requests the remaining missing blocks from multiple peers, rather than just requesting each of the blocks from a single peer at a time. Bittorrent sends cancels to blocks which it has already acquired to keep the bandwidth from being wasted on redundant sends. All in all, not much bandwidth is being wasted, because the end of the file is usually downloaded quickly with the help of endgame mode [2, 5].

3.5 Incentives in Bittorrent

As Bittorrent does not have any central resource allocation, the responsibility to maximize the utilization of transfer rates falls to the peers. To assure that the file distribution is efficient and that there is a certain level of fairness in the peer cooperation, Bittorrent utilizes a variation of the tit-for-tat as an incentive mechanism. The basic idea behind tit-for-tat is that if both participating parties in a transaction cooperate with each other, both will be able to gain the most out of the participation in the long run. Bittorrent tries to achieve this with various choking algorithms. [5]

3.5.1 Choking algorithm

The choking algorithm encourages peers to offer higher upload rates to obtain better download rates, in return. When a peer is not cooperating with a certain peer, it is choking, and when a peer cooperates with a certain peer it is unchoking. As each peer in Bittorrent always has a certain number on peers unchoked, the default being 4, the problem that the chocking algorithm needs to address is which peers to unchoke [5]. The choking algorithm decides which peers to unchoke solely based on download rates. So each peer uploads data to only 3 peers which provide it with the highest download rate. The higher upload rate a peer offers, the higher download rate it will most likely receive from others. This encourages peers to keep their upload rates high, so that they achieve better performance [11]. To prevent the situations where resources are wasted by rapidly choking and unchoking peers, Bittorrent peers use a ten-second interval between recalculations of who they want to choke and unchoke. Ten seconds is a long enough time for TCP to increase the transfer rates to maximum and the Bittorrent to determine whether some other peers have higher upload speeds available compared to the uploads speeds that the 3 current peers are offering. If better peers are found, they are unchoked and the 3 currently used peers are choked, and thus replaced by the new better-performing peers [5]. The number of the unchoked peers can be changed from the Bittorrent client application's settings.

3.5.2 Optimistic unchoke

The fourth peer slot is used to discover new unused connections that might offer even better transfer rates than the ones that are being used. In a situation where peers would just upload to peers that provide them with the best download rates, it would be impossible for new peers to join the swarm, because they have nothing to share. To address this problem every Bittorrent peer has always a single optimistic unchoke. An optimistic unchoke is a peer that is unchoked regardless of the download rate it is able to offer. This way new peers are able to acquire pieces of the file and start contributing to the

upload process themselves, and maybe offer better transfer rates to other peers [11]. The peer that is chosen as the optimistic unchoke is changed every 30 seconds. Thirty seconds should be enough time for the peer to increase its download process to full capacity and require pieces from the seeding peer and the download to reciprocate. [5]

3.5.3 Anti-snubbing

In some situations a peer might become choked by all of the peers it was previously downloading from. In that kind of scenario, the peer will continue to receive extremely low download rates until its optimistic unchoke finds better peers for it. To address this problem, in Bittorrent a peer is assumed to be "snubbed" by other peers after it has not received any pieces from them for over 60 seconds. After a peer notices that it has been "snubbed" by other peers, it stops to upload to those particular peers, except as an optimistic unchoke. In this manner there is a high probability that the peer will be chosen as an optimistic unchoke for those peers, which will cause the download rates to recover much faster when they slowdown. This method is called anti-snubbing and it is an exception to the single optimistic unchoke rule. [5]

3.5.4 Third party incentives

Bittorrent files are usually published through some centralized websites that consist of listings of the torrent files with HTML links to the Tracker, such websites are for example filemp3.org and torrentbytes.net. Often, the operators of such websites use different methods to further encourage cooperation between peers, in addition to the incentives that are used in the Bittorrent protocol itself. One of the most popular methods is the use of sharing -ratio enforcement. The sharing -ratio is calculated by dividing the amount of data uploaded by the amount of data downloaded. The main idea is to encourage peers to be above a minimum sharing -ratio value. The websites that utilize sharing -ratio enforcement usually maintain long-term records of its users download and upload behavior, and prevent the users that are below the certain

sharing -ratio to access new torrent material in the website. Some websites might even kick out a user who has had a low sharing -ratio too long. The sharing -ratio enforcement is usually implemented by requiring the users to register and login to the website before they can access and download the torrent files. By registering to the website, the users are linking their identity with their download activity, thus sacrificing their user anonymity. This method clearly encourages users to seed after their download has finished or even to upload files of their own to maintain the sharing -ratio above the required threshold. This way the mechanism provides an indirect incentive for seeding, as well as direct incentive not to freeride (download without uploading). The required sharing -ratio varies between different torrent websites, but the optimistic share -ratio is, of course, over 1, meaning that the peer has uploaded more data than it has downloaded [11].

Many of the websites that host torrent files offer the opportunity to follow the newly published files through Really Simple Syndication (RSS) feeds, one such website is thepiratebay.org. Broadcatching is a technology that combines RSS with Bittorrent protocol to create an automated content delivery system [1]. With Broadcatching the user is able to subscribe to an RSS feed of a website that publishes torrents and indicate which type of files he/she is interested in, and as soon as the files are made available in the website the Bittorrent client application starts to download them automatically. Because of the automated characteristic of Broadcatching, it can increase the times that the user spends as a seeder, because the user might leave the Bittorrent client application open for longer periods, while waiting for a specific file to be made available in a website. This results in creating more availability and cooperation in the Bittorrent network [12].

4 Differences between Bittorrent and tradition client-server schemes

The main difference between the Bittorrent protocol and traditional client-server models, such as File Transfer Protocol (FTP) and HyperText Transfer Protocol (HTTP) is that in FTP and HTTP, there is a central server that sends the entire file to each client that requests it, the clients only speak to the server and never to each other (see Figure 4.1) [13]. In Bittorrent, each client participating in the download process acts as a server and a client at the same time, in this manner the file is being uploaded from several sources (see Figure 4.2).

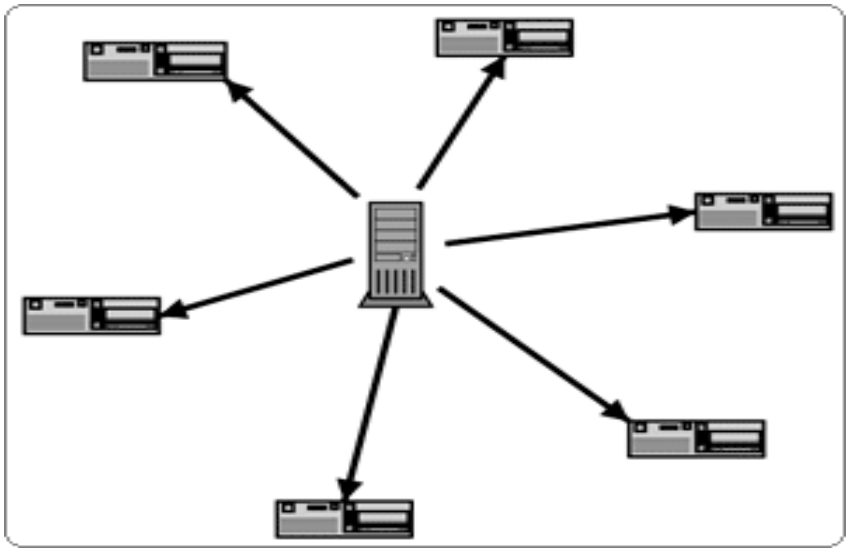

Figure 4.1 This is an example of simplified client–server implementation, in which one central server sends the entire requested file to each peer independently. [10]

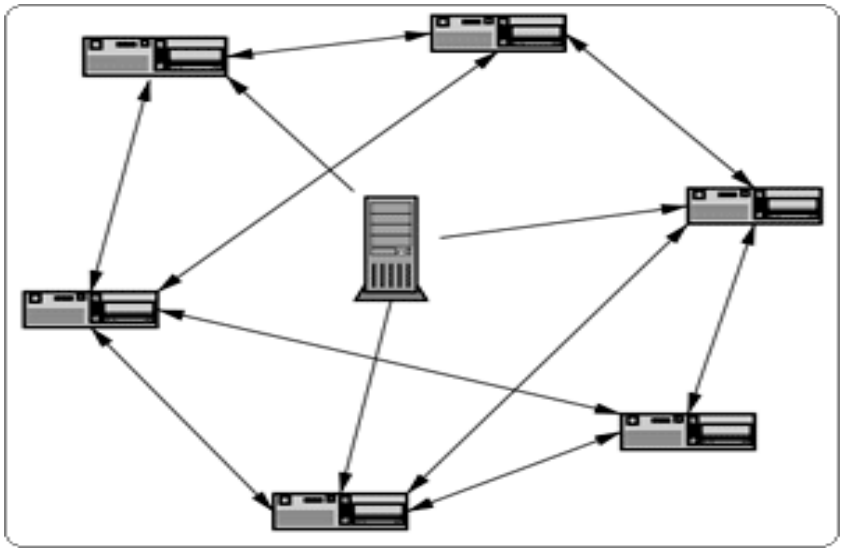

Figure 4.2 This is an example of Bittorrent protocol implementation, in which all the participating peers contribute to the distribution of the file, by acting as a client and a server at the same time. [10]

The main advantages of traditional client–server protocols are that they are easy to set up and the files are always available, because the servers are usually dedicated to the task of only hosting the files, are always on and connected to the Internet. However, the client-server models have great difficulties handling files that are large or popular, or both. The reason for this is that it takes a huge amount of bandwidth and server resources to distribute such a file, because the server has to transfer the entire file to each user that is requesting it. The problem has been partially solved by implementing several servers that host the same files, thus creating load-balancing between the servers, but this method is difficult and expensive to set up [13].

When using Bittorrent in a situation where the file that is distributed is large and popular, the available bandwidth scales linearly with the number of peers,

meaning the transfer rates should become higher as the number of downloaders increases. One famous example of such situation is when RedHat 9 Linux distribution ISO images were made available for public; The RedHat's main FTP server, and many of its mirrors were unable to cope with the high demand, and the servers provided slow transfer rates or were even in some occasions unconnectable for several weeks after the release. However, when the files were made available by using Bittorrent, it was possible for the downloaders to achieve several hundred Kb/s transfer rates. Although there is no minimum size limit for the files, Bittorrent is mostly used to distribute large files. The most significant reason for this is the time and effort that it takes to make a file available in Bittorrent; the Torrent file needs to be generated, added to the Tracker, and made available in some website through HTTP links. This usually means, in terms of time taken, that it is more practical to use Bittorrent to share few large files, rather than a large number of smaller files. [4]

4.1 Transfer rates

As both HTTP and FTP use a method where an entire file is transferred from a single source, it is hypothesized that Bittorrent would take less time to transfer a file when multiple simultaneous downloads are happening, because the sharing is done amongst all the participating clients. An earlier piece of research [3] studying the transfer time differences between Bittorrent, HTTP and FTP validates the fact that as the number simultaneous downloads increases, so does the efficiency of Bittorrent. Table 1 shows the results of a test, where a 1GB file was transferred using Bittorrent, HTTP and FTP. All the machines taking part in the study had network connections of the same speed. The tests were conducted incrementally, so that in the first test the host server transferred the file to a single machine and in the second test the file was transferred to two machines, and so on, until the last test where the file was transferred to 11 other machines simultaneously. Table 1 shows that the download times are shorter for both FTP and HTTP when the server-client ratio is 1-to-1 and 1-to-2. However, as the number of downloading clients is increased to 3, Bittorrent becomes the fastest way to distribute the file. In the situation where there are 11

simultaneous machines downloading, the Bittorrent transfer takes approximately one-third of time compared to the FTP and HTTP, which in terms of time equals to approximately 2 hours [3]. This clearly shows that it is preferable to use Bittorrent, when the file is expected to be popular, and FTP or HTTP when the file is not expected to have many simultaneous downloads.

Table 1. The 1 GB file transfer times for each protocol in milliseconds (ms) with different server - client ratios. [3]

Ratio	1GB BT	1GB HTTP	1GB FTP
1-to-1	1073493	999343	1018997
1-to-2	2029849	1988906	2011463
1-to-3	2689217	2968875	3003322
1-to-4	2908993	3949655	3991422
1-to-5	2021227	4935430	4978921
1-to-6	2620776	5911176	5967121
1-to-7	2591917	6893301	6958753
1-to-8	3238437	7910424	7982288
1-to-9	3509326	8912535	8983352
1-to-10	3501215	9924880	9992839
1-to-11	3657008	10958656	11023011

4.2 Webseeding

Webseeding was implemented in 2006 as an extension to the Bittorrent client applications to provide the ability to cooperate with HTTP and FTP sources [1]. Many of the websites that offer file downloads using Bittorrent, might also provide HTTP or FTP URLs for the same files. As a result, webseeding was

created to utilize both resources at the same time. When webseeding is enabled in the Bittorrent client application, the user can download the file pieces from both Bittorrent swarm and HTTP/FTP server and compile the complete file from the pieces (provided that the same file is made available from both sources). The advantage of webseeding is that the HTTP or FTP server acts as a permanent unchoked seed. In this manner there will always be a peer to connect with; so that the download can start immediately, but in a case of multiple simultaneous downloads the upload strain is still divided among all the participating downloaders. Also, the users who do not have webseeding enabled in their Bittorrent client application would benefit from the peers that are sharing the pieces that were originally obtained from the HTTP/FTP server [14]. The web seed URLs can be added or viewed in the advanced interleaf of the torrent file properties.

5 Attacks and abuses of the Bittorrent protocol

Bittorrent suffers from various attacks and abuses that are utilized by people who have very different motivations; some are trying to degrade the performance of the protocol, while some are trying to boost it. The attackers are usually split in to two groups; end-users and anti-P2P companies. An example of such anti-P2P company is MediaDefender, Inc [15]. While the methods used by end-users usually consist of ways to improve download rates and avoiding uploading, the anti-P2P companies aim to impair to distribution of certain files. Also, it has been noted that some Internet Service Providers (ISPs) are trying to block their customers from uploading data using Bittorrent protocol to limit the strain that the protocol might cause to their bandwidth resources [16].

5.1 Attacks by Anti-P2P companies

Just like other previous popular P2P file sharing systems, such as Kazaa and eDonkey, Bittorrent has started to draw the attention of organizations such as the Motion Picture Association of America (MPAA) and Recording Industry Association of America (RIAA) that try to fight against the illegal file distributions

of copyrighted material. In the previous years, MPAA and RIAA have managed to dismantle P2P file sharing companies, such as Kazaa and MetaMachine (developer of eDonkey). These organizations succeeded in their goal by filing numerous law suits against the P2P- file sharing companies, tracking and suing the users, and even launching large-scale Internet attacks against the P2P systems themselves. The Internet attacks were carried out by anti-P2P companies working on behalf of the RIAA, MPAA and specific record labels and movie companies [17].

As Bittorrent is one of the most popular P2P file distribution protocols nowadays, and an especially effective way to distribute large files, it has become a very popular means for distributing illegal copyrighted material. Unlike Kazaa and MetaMachine, Bittorrent is nothing more than a protocol, and the swarms and clients in it are not controlled by any small set of companies which can be targeted for lawsuits. In Bittorrent the people who host the Trackers and administer the websites that have the .torrent -files are susceptible to lawsuits [17]. An example of such situation happened on December 19th 2004, when Suprnova, the largest torrent locator website at the time was closed down after receiving legal threats [18]. These days it is more difficult, if not impossible, to stop illegal file distribution in Bittorrent using lawsuits, because of the decentralization of trackers using DHT, Local Peer Discovery and Peer Exchange. As a result, the music and film industry have started to hire anti-P2P companies to launch Internet attacks against the Bittorrent protocol itself to hinder the distribution of their products [17]. The most common attacks that the anti-P2P companies utilize nowadays consist of fake-block attack and uncooperative-peer attack [15, 17]. Anti-P2P companies also use a method where they upload files to torrent websites that appear authentic, but in fact are corrupted. This method is not very effective, because the users in the websites usually warn each other not to download these files, and in most torrent websites the administrators go through these corrupted torrents as a daily job and delete them before they are downloaded [9].

5.1.1 Fake-block attack

A fake-block attack is used to prolong the download times of a certain file by wasting the download bandwidths of the users who are trying to obtain the file in question. As previously mentioned, in Bittorrent the distributed file is cut into pieces, where each piece is usually 256 Kbytes. Each piece is further cut in to blocks of the size of 16 Kbytes. When downloading a piece, the client requests different blocks from different peers. In the fake-block attack the attacker contacts the tracker and joins a swarm. When it is in the swarm it advertises to all peers that it has a large number of pieces of the file that is being distributed. As the victim peer receives this advertisement it contacts the attacker peer and requests a block from it. Instead of sending the authentic block, the attacker sends a fake one. After downloading all the blocks in a piece from the attacker and other legitimate peers, the victim peer performs a hash check across the entire piece. Due to the fake block in the piece the hash check fails, and the victim peer has to download the entire piece (16 blocks) again, thus delaying the completion of the file. If the victim peer chooses to download any of the blocks again from the same attacker or any other attacker while downloading the piece, the download is further delayed. The fake-block attack is an efficient way to hinder the download processes, because the attacker only needs to send a block of 16 Kbytes to make the victim peer waste 256 Kbytes of download bandwidth. [17]

5.1.2 Uncooperative-peer attack

In the uncooperative-peer attack, the objective is to waste the time of the victim peers as much as possible. In the attack, the attacking peer joins a swarm and establishes TCP connections with victim peers. However, it never provides any blocks, fake or real, to the victim peers. A common version of the attack is called chatty peer attack. In the chatty peer attack, the attacking peer speaks Bittorrent protocol with the victim peer, starting with the handshake where it introduces itself, and followed by a bitmap message where it advertises the number of pieces it has for the given file. When the victim peer requests one or

more blocks, the attacking peer does not reply to the request. Instead the attacking peer starts the process from the beginning by sending the victim peer the handshake and bitmap messages again, and repeats the same process over and over again until the victim chooses to download the blocks from other peers. This way the victim peer wastes considerable amount of time by dealing with the attacking peer, when it could have been downloading the blocks from other legitimate peers. Uncooperative-peer attacks are most effective when a significant fraction of victim's neighbors in a swarm are uncooperative. [17]

5.1.3 Effects of the anti-P2P attacks

A study [17] about the effects of fake-block attack and uncooperative-peer attack in Bittorrent shows that the anti-P2P companies are successful in prolonging the download times of files by attacking them with these two attacks. The download times are longer especially when using residential broadband, but the extent of the prolonging is not really enough to make a significant difference. The study shows that it takes approximately 50 % more time to download a file that is being attacked than it would normally, which is not a lot when considering the bandwidth speeds of current Internet connections. So, to achieve their goal in bringing down the distribution of certain illegal files the anti-P2P companies should invest a lot more resources to the attacks, so that the download times would become unbearable. However, even if the download times would triple, it still might not be enough, as the Internet connections are becoming faster all the time, and the Bittorrent users are considered to be fairly patient and download files overnight or in background [17].

5.1.4 Countermeasures against anti-P2P attacks

At the moment, the only way to protect against anti-P2P attacks is to use IP blocklists (also known as IP blacklists). IP blocklists are updated almost daily and they contain all the known IP addresses of anti-P2P companies, peers and trackers that are used to attack Bittorrent and distribute corrupted data. The function of the lists is to block incoming and outgoing connections based on IP

addresses. The lists can be downloaded from various websites, such as www.blocklistpro.com and used in conjunction with third party applications, such as PeerBlock. Also, the Bittorrent client application µTorrent has an IP filtering function implemented in it and can utilize IP blocklists by itself. Although these blocklists offer some level of protection, they will not be able to stop all the attacks, because the anti-P2P companies are all the time using new IP addresses to launch the attacks and the blocklists can only be updated after a such an attacking IP address is identified [19, 20].

5.2 End-user abuses

The end-user abuses in Bittorrent usually revolve around users trying to obtain better download rates or users refusing to upload. Although the tit-for-tat mechanisms in Bittorrent is an efficient way to assure fairness in the file transfer process, there are still some users who succeed to bend the rules the in hopes of achieving faster download rates and minimum upload rates. The greatest problem in Bittorrent, much like in many other P2P file distributing systems, is freeriding.

A freerider is a user who downloads data, but does not upload it. Freeriding can happen for multiple reasons: a user might have limited uplink bandwidth and wishes to save it for other critical tasks, or a user might be downloading illegal copyrighted material and thinks that uploading it is much riskier, or user's firewall might restrict the ability upload. Whatever the reason might be, freeriders present the threat that if the Bittorrent network becomes overrun by users who do not upload anything, the network can suffer from system-wide performance degradation and become unusable.

The tit-for-tat mechanism in Bittorrent works on the principle that the user achieves better download rates by offering high upload rates to others, which obviously discourages freeriding, but other studies [12, 21] have shown that in a swarm where there are a lot of seeders, the freeriders can actually achieve higher download rates than the peers that follow the rules of tit-for-tat. This works as a huge incentive to freeride, but the downside is that when the number

of freeriders becomes higher than the number of compliant peers, everybody suffers substantially from performance degradation, because the upload strain becomes overwhelming for the small number of compliant users. To reduce the number of freeriders, Bittorrent communities have started to use sharing -ratio as a way to discourage freeriding. For the sharing -ratio enforcement to be effective, the websites that host the torrent files need to require the users to register and login before they can access the files. In this way, their downloading and uploading behaviors can be monitored and any user who is freeriding can be detected quickly and kicked out, and thus preventing them from accessing the website's Tracker and decreasing the performance of all the other participating users. A study [12] about the influences on cooperation in Bittorrent communities validates that using sharing -ratio as an additional mechanism to enforce cooperation among peers really does reduce freeriding. The study also shows that the websites that require the users to register and login before accessing the torrent files have much higher seeding levels, compared to the websites that offer the files without any restrictions.

5.3 Bittorrent Blocking

As Bittorrent is an extremely popular file sharing system, its traffic has started to account for a large and ever growing fraction of the traffic that traverses the Internet. The resulting increase in the Internet traffic is raising the cost of transit for the ISPs, and many of them have started to implement strategies to reduce the amount of Bittorrent traffic generated by their customers [16]. The ISPs utilize middleboxes, such as traffic shapers, blockers, and firewalls in their network to rate-limit the customers bandwidth consumed by Bittorrent. Moreover, some ISPs have been even found to block the Bittorrent generated traffic entirely [22, 9]. An example of such ISP has been Comcast, an ISP based in America, which exclusively blocked only Bittorrent- generated traffic in their network, and not any other protocols. Comcast was forced to stop the blocking in 2008 by the order of Federal Communications Commission (FCC), which concluded that the company's network management practices were unfair [9].

The ISPs block the traffic by injecting forged TCP reset (RST) packets into the traffic flows. When the receiving end of the Bittorrent traffic receives these RST packets, it terminates the connection immediately. A piece of research [16] about ISPs blocking Bittorrent traffic shows through various tests that the ISPs primarily block only Bittorrent uploads and rarely interfere with the downloads. The research also shows that the traffic is blocked mostly by identifying Bittorrent messages from the traffic flow with the use of deep packet inspections [16].

5.3.1 Countermeasures against Bittorrent blocking

Users who are suffering from slow transfer rates can use various software programs, such as Measurement Lab (also known as M-Lab) to test their connection and see if their Bittorrent traffic is being blocked by the ISP [22]. Bittorrent users can also try to avoid becoming blocked or throttled by encrypting their traffic. To encrypt their traffic, the users can either pay to use third-party VPN services, which usually charge monthly payments for their services, or the users can utilize the encryption function in Bittorrent clients. Nowadays all of the most popular Bittorrent clients, such as Mainline (Bittorrent), Azureus and µTorrent offer the possibility to use protocol header encryption on all the outgoing traffic and only accept encrypted incoming traffic. The encryption can be turned on from the client application's options (see Figure 4.1). When the encryption is enabled, it is more difficult for the ISPs to distinguish Bittorrent traffic from other traffic and block it, but it does not help against more aggressive forms of traffic interference, such as ISPs using Sandvine applications [9, 23, 24]. Also, using encryption takes more CPU time and prevents the Bittorrent client from accepting connections from peers who do not use encryption, if the "Allow incoming legacy connections" checkbox is not ticked in the protocol encryption menu.

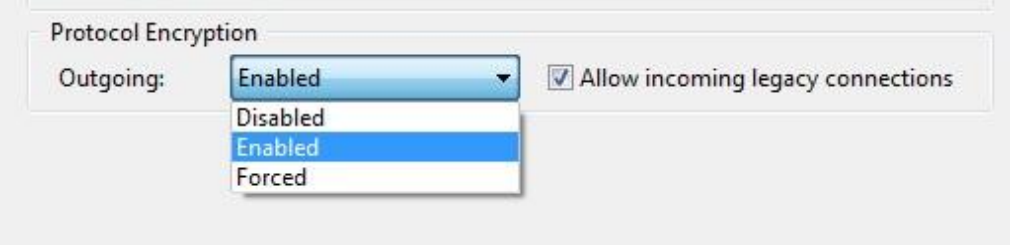

Figure 4.1 An illustration of the protocol encryption options in Bittorrent (Mainline) client application.

New, better technologies to hide traffic from ISPs are being developed all the time, and one such promising technology is BitBlinder. BitBlinder works for both Bittorrent and normal web traffic, and it aims to make Bittorrent transfers anonymous and allow users to avoid the most common restrictions and filters that the ISPs might have implemented on their Internet access. The BitBlinder provides these functions through its own P2P network, where every user who is participating in it contributes their own bandwidth to proxy other users' data. In practice, BitBlinder works so that in a situation where a user, for example, wants to visit a website, the BitBlinder asks the users' peer in the network to obtain the page and return it to the user, instead of asking for the page directly from the website itself. As a result, the website will not even know that the user exists, because it received the request from the peer. BitBlinder always encrypts the connections to its peers, so that that eavesdroppers can not find out what information the user sent.

To add more security to the system, the BitBlinder makes the users' requests go through 3 peers, instead of 1, (see Figure 4.2). In this way no one will know the identity of the user, because each peer only knows about the previous peer they talked to and the next peer. For the traffic to exit the BitBlinder network to the wider Internet, some users need to act as exitpoints, which means that the

traffic leaving the network will be always associated with the IP address of the exitpoint computer. In a situation where a user acting as an exitpoint is accused of downloading illegal material, the user can reasonably claim that traffic coming from its computer did not originate from it, even if it did, because of the proxying characteristic of BitBlinder. The same rules apply to Bittorrent traffic that traverses the BitBlinder network. The BitBlinder technology is still in its beta testing phase, so only the future will show whether or not it will be able to work around the different restrictions that ISPs have for Bittorrent traffic. [9, 25].

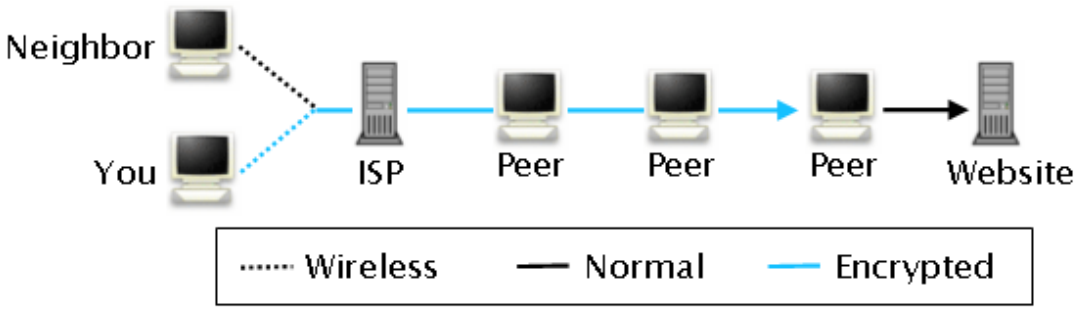

Figure 4.2 Illustration of how the identity of the traffic originator is concealed from the destination by using peers as proxy servers in BitBlinder. [25]

6 Summary

Since its release, the Bittorrent protocol has become one of the most popular and controversial P2P file distribution protocols used today. Its effectiveness in distributing large files has made it a very popular way of distributing illegal copyrighted material, which has caused the protocol to receive a lot of negative attention. Through the in-depth presentation of the protocol's functions, this thesis shows that Bittorrent is a superior file distribution method in high-demand networks, mainly because of the tit-for-tat strategy implemented in it. The tit-for-tat mechanism does not only enforce fairness in file transfers, but it also builds

robustness against freeriding and other forms of attacks. The thesis also shows that the Bittorrent protocol has evolved a lot from the first version that was released in 2001, clearly heading from the centralized system characteristics to the decentralized system characteristics, which indicates that in the future the use of trackers might be abandoned entirely.

The findings in this thesis are mainly based on theoretical information, so whether or not Bittorrent will evolve to be a fully decentralized system will be determined in the future, when such implementations are tested in practice. As for future studies, the information presented here could be used as background information for a study of the functionalities and implementation of a fully decentralized Bittorrent network that would make the torrent indexing websites and trackers obsolete.

References

[1] Miller, F.P., Vandome, A.F., McBrewster, J. Bittorrent (protocol). Mauritius, 2009

[2] Bittorrent, [www- document]. Available at: http://www.bittorrent.com. (Referred: 3.3.2010)

[3] Lincoln, S., "Network File Distribution with the BitTorrent Protocol", [PDF–document]. Available at: http://cs.winona.edu/CSConference/2007proceedings/lincoln.pdf. (Referred: 15.2.2010)

[4] Nicoll J.R., Bateman, M., Ruddle, A., Allison, C. "Challenges In Measurement and Analysis of the BitTorrent Content Distribution Model", [PDF–document]. Available at: http://distsyst.cs.st-andrews.ac.uk/btpaper.pdf. (Referred: 27.2.2010)

[5]Cohen, B., "Incentives Build Robustness in BitTorrent", [PDF–document]. Available at: http://www.bittorrent.org/bittorrentecon.pdf (Referred: 22.1.2010)

[6] Neglia, G., Reina, G., Zhang, H., Towsley, D., Venkataramani, D., Danaher, J., "Availability in BitTorrent Systems", [PDF–document]. Available at: http://wwwcs.umass.edu/~arun/papers/

[7] TorrentEditor, [www- document]. Available at: http://torrenteditor.com/faq.php. (Referred: 26.2.10)

[8] Softpedia, [www- document]. Available at: http://news.softpedia.com/news/BitTorrent-Magnet-Links-Explained-132536.shtml. (Referred: 23.2.2010)

[9]] Torrentfreak, [www- document]. Available at: http://torrentfreak.com/. (Referred: 26.2.2010)

[10] Bittorrent, [www- document]. Available at:http://www.bittorrent.org. (Referred: 10.2.2010)

[11] Weidong, L., Dongsheng, P., Chuang, L., Chen, Z., Song, J., "Enhancing tit-for-tat for incentive in BitTorrent networks", Peer-to-Peer Networking and Applications, Vol. 3, Number 1, pp 1-9, March 2010

[12] Andrade, N,. Mowbray, M., Lima, A., Wagner, G., Ripeanu, M., "Inuences on Cooperation in BitTorrent Communities", [PDF– document]. Available at: http://conferences.sigcomm.org/ sigcomm/2005/paper-AndMow.pdf. (Referred: 27.2.2010)

[13] Dessent, B., "Brian's Bittorrent FAQ and guide", [www- document]. Available at: http://dessent.net/btfaq/#what. (Referred: 9.2.2010)

[14] Burford, M., "Web Seed - HTTP/FTP Seeding", [www- document]. Available at: http://www.bittorrent.org/beps/bep_0019.html. (Referred: 22.2.2010)

[15] Mediadefender, [www- document]. Available at: http://www.mediadefender.com. (Referred: 17.2.2010)

[16] Dischinger, M., Mislove, A., Haeberlen, A., Gummadi, K., "Detecting BitTorrent Blocking", [PDF– document]. Available at: http://www.mpi-sws.org/~gummadi/papers/08_imc_blocking.pdf. (Referred: 11.2.2010)

[17] Dhungel, P., Wu, D., Schonhorst, P., Ross, K.W., "A Measurement Study of Attacks on BitTorrent Leechers", [PDF– document]. Available at: http://www.cs.toronto.edu/iptps2008/final/

47.pdf. (Referred: 28.2.2010)

[18] Ingram, M., “Suprnova.org: The story of a legend”, [www-document]. Available at: http://www.slyck.com/news.php?story=1177. (Referred: 16.2.2010)

[19] BlocklistPro, [www- document]. Available at: http://blocklistpro.com/faqs/p2p-ip-block-lists.html. (Referred: 17.2.2010)

[20] PeerBlock, [www- document]. Available at: http://www.peerblock.com/. (Referred: 17.2.2010)

[21] Sirivianos, M., Park, J.H., Chen, R., Yang, X., “Free-riding in BitTorrent Networks with the Large View Exploit”, [PDF– document]. Available at: http://research.microsoft.com/en-us/um/redmond/events/iptps2007/papers/SirivianosParkChenYang.pdf. (Referred: 25.2.2010)

[22] Calore, M., “New Google Tools Determine if Your ISP Is Blocking BitTorrent”, [www-document]. Available at: http://www.wired.com/epicenter/2009/01/new-google-tool/ (Referred: 19.2.2010)

[23] filesharingz.com, “Bittorrent encryption myths”, [www-document]. Available at: http://

/bittorrent-encryption-myths.php. (Referred: 19.2.2010)

[24] Sandvine, [www- document]. Available at: http://www.sandvine.com/. (Referred: 19.2.10)

[25] BitBlinder, [www- document]. Available at: http://www.bitblinder.com/. (Referred: 19.2.10)